

APPLICATION FOR LETTERS PATENT OF THE UNITED STATES

SPECIFICATION

To all whom it may concern:

Be It Known That We, Colin Luck and Pierre Colin, have invented certain new and useful improvements in **STORING AND DELIVERING PROGRAM CODE FOR MANIPULATION OF A USER-DEFINED DATA TYPE**, of which I declare the following to be a full, clear and exact description:

STORING AND DELIVERING PROGRAM CODE FOR MANIPULATION OF A USER-DEFINED DATA TYPE

5

Background

As rapid advancements in technology have brought about dramatic changes in the distribution of information in recent years, the managers of commercial data repositories have seen rapid increases in the amounts and types of information that they are asked to store. The growing popularity of multimedia information, for example – including items such as photographic images, video and audio clips, and animation clips – has forced the manufacturers and owners of data repositories to overhaul the ways in which they manage the information they keep. Relational database systems, which have traditionally been used to manage only information that is easily divided into a row-and-column format, now must handle more complex data types that do not conform readily to relational data structures.

Manufacturers of relational database systems have for years attempted to accommodate these more complex data types by treating them as “objects,” or large groups of data that each can be stored and treated as a single entity. Quite often, a database system stores separately in its relational structure the data that makes up an object – such as a binary bitmap that forms a photographic image – and a set of descriptive data, or “metadata,” that describes the object.

In general, however, the database system does not allow for easy manipulation of object data on the client side. For example, the vast majority of database systems do not support object-oriented design of user-defined data types (UDTs) on client systems in traditional database-query languages, such as the Structured Query Language (SQL), instead requiring more time-consuming coding through programming languages such as C++. Even the most sophisticated of relational database systems present object data, including UDTs, to the client in ways that severely limit the manner and environment in which the client can use that data. These limitations include an inability to move object data among non-compatible hardware platforms and a lack of support for multiple

versions of a UDT. Database systems also do not allow the execution of methods on UDTs on the client side. Client systems that modify the data of a UDT cannot return the modified data to the database system.

5

Summary

A database system carries out a technique for use in delivering data of a user-defined type to a requesting system. The system first receives a request from the requesting system for data of the user-defined type. The system then identifies a storage
10 location for the requested data, as well as a storage location for program code designed to allow manipulation of the requested data on the requesting system. The system then retrieves both the requested data and the program code from their respective storage locations and delivers both the requested data and the program code to the requesting system.

15 Other features and advantages will become apparent from the description and claims that follow.

Brief Description of the Drawings

20 FIG. 1 is a schematic diagram of a database system and a client computer system.

FIG. 2 is a diagram of a relational table in a database system.

FIG. 3 is a flow chart of a technique for use in storing program code for use in manipulating data of a user-defined type along with the data.

FIG. 4 is a flow chart of a technique for use in delivering program code for use in
25 manipulating data of a user-defined type along with the data.

FIG. 5 is a schematic diagram of a computer system.

FIG. 6 is a schematic diagram of a database system.

Detailed Description

Described below is a technique that allows easy client-side manipulation of data of a user-defined type (UDT). The creator of the UDT, on creating the UDT, also creates
5 program code that allows manipulation of data of the user-defined type and then stores this code in a database system along with the data of the user-defined type. When a client system requests data of the user-defined type, it also receives the program code, which typically is written in the same language as the UDT, typically the Structured Query Language (SQL), put forth by the American National Standards Institute (ANSI). The
10 client system is able to store the program code so that it does not have to download the code each time the UDT is accessed. Storing the program code in this manner allows easy manipulation of UDTs across platforms and without distribution of software through traditional channels.

FIG. 1 shows a system in which a client computer 100 interacts with a database
15 system 110 through a computer network 120 in the storage and retrieval of data. The client computer 100 includes a processor and a data-storage facility that together allow the computer 100 to execute an application program 130. The application program 130 allows a user of the client computer 100 to create or to retrieve and then manipulate data that is stored in the database system 110.

20 The database system 110 includes a host computer system 140 connected to a data-storage facility 150 that stores a very large amount of data. As described in more detail below, the database system 110 typically includes a relational database-management system (DBMS) that stores data in relational tables. Included in this data is object data 160 – such as multimedia data and data of a user-defined type (UDT) – that does not lend
25 itself readily to the structure of a relational database. The data 160 of the UDT is stored in the data-storage facility 150 along with program code 170 that allows manipulation of the data, typically somewhere outside the relational table structure.

The program code 170 is written by the creator of the UDT and is designed to allow
30 execution of data-manipulation operations – such as user-defined methods – that the creator wishes to perform on the data of the UDT. For example, for a UDT that includes

geographical-location information derived from a map, the creator might include a manipulator that allows the calculation of distance between two points on the map. Instead of requiring systems that would access the map data to have previously stored the program code needed to calculate distance, the database system stores this code along
 5 with the data itself. On receiving a request for the geographical-location data, the database system delivers the distance-calculation code to the requesting system along with the requested data.

FIG. 2 shows one way that the data 200 of a user-defined type and the corresponding program code 210 needed to manipulate that data are managed in a
 10 relational table 220 of a database system. The table 220 includes rows and columns that organize data according to some predefined relational structure. The data 200 of the UDT typically is not susceptible to storage in this relational structure, so the database system stores this data 200 elsewhere in the system. The table 220 instead stores metadata 230₁, 230₂ that describes the data 200 of the UDT, along with a pointer 240₁ to
 15 the storage location of the data 200.

In addition to the pointer 240₁ to the UDT data 200, the table 220 also stores a pointer 250₁ to the storage location of the program code 210 that accompanies the UDT data. As with the UDT data 200, the program code 210 itself is stored somewhere in the database system other than in the relational table 220. When another computer system
 20 requests the data of the UDT, the database system accesses the table 220 to find the data needed to answer the request and to identify the corresponding program code.

FIG. 3 shows a technique for use in a database system in storing data of a user-defined type along with the program code needed to manipulate that data. On receiving a request to store data of a user-defined type (step 300), the database system receives the
 25 data along with information describing the UDT (step 310). The database system also receives the program code that is needed to manipulate the data of the UDT (step 320). The database system then stores both the UDT data and the program code in one of its data-storage facilities (step 330). The system also inserts at least some of the data describing the UDT into one or more relational tables (step 340) and, along with this

data, inserts pointers to the storage locations of the UDT data and the corresponding program code (step 350).

FIG. 4 shows a technique for use in the database system in retrieving the UDT data and the corresponding program code. On receiving a request for data of the user-defined type (step 400), the database system accesses its relational tables to identify the location of the data needed to answer the request (step 410). The system also searches the tables for the location of the corresponding program code, if any (step 430). On finding pointers to the data and the program code, respectively, the system retrieves the UDT data (step 420) and the corresponding code (step 440) from their storage locations and delivers both to the requesting system (step 450).

In some embodiments, the request for data includes information indicating whether the requesting computer has previously requested data of the user-defined type and, if so, which version of the program code the computer previously received. On receiving the request, the database system assesses whether the requesting computer has indeed requested that type of data (step 450). If so, the database system assesses which version of the program code the requesting computer received before (step 460). If that version matches the current version of the program code, the database system sends only the requested data without the program code (step 470). Otherwise, the system sends both the requested data and the corresponding code.

FIG. 5 shows a computer system 500 suited for use both as a client computer and as a host computer. In general, the computer 500 includes one or more processors 505, one or more temporary data-storage components 510 (*e.g.*, volatile and nonvolatile memory modules), one or more persistent data-storage components 515 (*e.g.*, optical and magnetic storage devices, such as hard and floppy disk drives, CD-ROM drives, and magnetic tape drives), one or more input devices 520 (*e.g.*, mice, keyboards, and touch-screens), and one or more output devices 530 (*e.g.*, display consoles and printers).

The computer 500 includes executable program code 535 that is usually stored in one of the persistent storage media 515 and then copied into memory 510 at run-time. The processor 505 executes the code by retrieving program instructions from memory in a prescribed order. When executing the program code, the computer receives data from

the input and/or storage devices, performs operations on the data, and then delivers the resulting data to the output and/or storage devices.

In some embodiments, the computer is a special-purpose computer that performs only certain, specialized functions. In other embodiments, the computer is a general-purpose computer programmed to perform the necessary functions.

FIG. 6 shows a data-warehouse system 600 that is suited for use in storing and retrieving the type of data and program code described above. In this example, the data-warehouse 600 includes a relational database management system (RDBMS) built upon a massively parallel processing (MPP) platform. Other types of database systems, such as object-relational database management systems (ORDBMS) or those built on symmetric multi-processing (SMP) platforms, are also suited for use here.

As shown here, the data warehouse 600 includes one or more processing modules 605_{1...Y} that manage the storage and retrieval of data in data-storage facilities 610_{1...Y}. Each of the processing modules 605_{1...Y} manages a portion of a database that is stored in a corresponding one of the data-storage facilities 610_{1...Y}. Each of the data-storage facilities 610_{1...Y} includes one or more disk drives.

The system stores data in one or more relational tables in the data-storage facilities 610_{1...Y}. The rows 615_{1...Z} of the tables are stored across multiple data-storage facilities 610_{1...Y} to ensure that the system workload is distributed evenly across the processing modules 605_{1...Y}. A parsing engine 620 organizes the storage of data and the distribution of table rows 615_{1...Z} among the processing modules 605_{1...Y}. The parsing engine 620 also coordinates the retrieval of data from the data-storage facilities 610_{1...Y} in response to queries received from a user at a mainframe or a client computer 630. The data warehouse usually receives queries in a standard format, such as the Structured Query Language (SQL) put forth by the American National Standards Institute (ANSI).

The text above describes one or more specific embodiments of a broader invention. The invention also is carried out in a variety of alternative embodiments and thus is not limited to those described here. Many other embodiments are also within the scope of the following claims.